

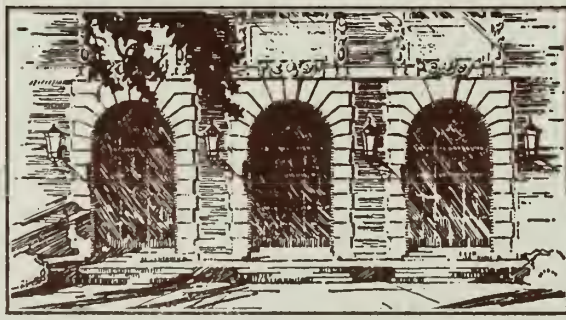
LIBRARY OF THE
UNIVERSITY OF ILLINOIS
AT URBANA-CHAMPAIGN

510.84

Il6r

no. 279-286

cop. 2



CENTRAL CIRCULATION BOOKSTACKS

The person charging this material is responsible for its renewal or its return to the library from which it was borrowed on or before the **Latest Date** stamped below. **You may be charged a minimum fee of \$75.00 for each lost book.**

Theft, mutilation, and underlining of books are reasons for disciplinary action and may result in dismissal from the University.

TO RENEW CALL TELEPHONE CENTER, 333-8400

UNIVERSITY OF ILLINOIS LIBRARY AT URBANA-CHAMPAIGN

9/17/96

SEP 06 1996

When renewing by phone, write new due date below previous due date.

L162



Digitized by the Internet Archive
in 2013

<http://archive.org/details/adaptivelinearcl284ibar>

ADAPTIVE LINEAR CLASSIFIER
BY LINEAR PROGRAMMING

by

Toshihide Ibaraki and Saburo Muroga

September 27, 1968



DEPARTMENT OF COMPUTER SCIENCE · UNIVERSITY OF ILLINOIS · URBANA, ILLINOIS

Report No. 284

ADAPTIVE LINEAR CLASSIFIER
BY LINEAR PROGRAMMING

by

Toshihide Ibaraki and Saburo Muroga

September 27, 1968

Department of Computer Science
University of Illinois
Urbana, Illinois 61801

ABSTRACT

This paper discusses a linear classifier based on linear programming which is adaptive to a change in the set of input vectors. Different from linear classifiers discussed in the past, this linear classifier maintains the maximum reliability of its operation. A procedure of deriving an optimum structure of the linear classifier for a new set of input vectors is a modification of the ordinary simplex method and yields an optimum structure in a much fewer iterations than the determination of the structure by straightforward application of the ordinary simplex method.

The adaptive procedure then is extended to other cases such as a linear classifier which maintains the minimum number of erroneously classified input vectors. This is based on Gomory's algorithm for integer linear programming.

The feasibility and efficiency of our linear classifiers are computationally proved by some examples.

Index Terms

- Adaptive network
- Simplex method
- Linear programming
- Integer linear programming
- Gomory's algorithm
- Dual linear programming
- Linear classifier
- Reliability of adaptive network
- Input tolerance of adaptive network

1. Introduction

A linear classifier for pattern recognition has received much attention recently for its cognitive power with simple structure and also as a constituent of a more complex classifier [1], [2]. There are at least two types of linear classifiers distinguished by their design, i.e., the one designed by adaptive (learning) methods and the other designed by linear programming. In this paper we will discuss a linear classifier which is based on linear programming but which is augmented with adaptive capability.

An adaptive method was first discussed in conjunction with the learning capacity of neural networks [2]. It has some merits such as (1) simplicity of training rule, (2) economy of memory space and (3) flexibility, in other words, it can adjust itself easily according to the change of environment (pattern vectors which are being processed) by constantly applying the training procedure. If the given pattern vectors are not linearly separable, however, the successive changes in the structure of an adaptive classifier is not simple. And even if the patterns of a pattern classifier are separable, the concept of optimality is not incorporated in the adaptive methods.

Linear programming methods, on the other hand, guarantee the optimality of a given linear function of the network parameters such as the sum of all weights. [3], [4] The paper by Smith [4] shows that computation time for adaptive methods is less than that for the linear programming methods when the number of pattern vectors is much greater than the dimension of vectors, and that the linear programming methods are preferable otherwise. However all these papers deal only with the

static case in which only a single set of pattern vectors is given and do not deal with a general dynamic case when new pattern vectors replace some of the current vectors. Thus linear classifiers discussed by them cannot adapt to a changing environment.

Our objective of this paper is to show a linear programming method which has the capability of adaptation even in changing environment. In other words, our linear classifier can adjust itself to a changing set of pattern vectors, always keeping the optimum structure of the classifier and without increase of the size of storage space to be used. This paper will discuss also a linear classifier based on an integer linear programming method which minimizes the number of error vectors, even if current set of pattern vectors is not linearly separable, and which also adapts to a changing set of pattern vectors.

2. Problem Statement

We consider a linear classifier which separates M N -dimensional pattern vectors

$$\vec{\xi}^{(j)} = (\xi_1^{(j)}, \dots, \xi_N^{(j)}) \quad j=1, 2, \dots, M \quad (2.1)$$

into two categories A and B for which the output f of the classifier is assumed 1 and 0 respectively. The coordinates $\xi_i^{(j)}$ are real numbers. An objective in designing a linear classifier is determination of a weight vector

$$\vec{w} = (w_1, \dots, w_N) \quad (2.2)$$

and a constant value

$$w_0 \quad (2.3)$$

of real numbers which satisfy

$$\begin{aligned} \vec{w} \cdot \vec{\xi}^{(j)} + w_0 &\geq d & \text{if } f = 1 \\ \vec{w} \cdot \vec{\xi}^{(j)} + w_0 &\leq -d & \text{if } f = 0 \end{aligned} \quad (2.4)$$

$$j = 1, \dots, M,$$

where $\vec{w} \cdot \vec{\xi}^{(j)}$ is the inner product $\sum_{i=1}^N w_i \xi_i^{(j)}$ and $d > 0$ is called

the margin of the classifier*. The margin is provided to secure reliable

* Even if the value of the left hand side of (2.4) does not exceed d (does not reach $-d$) for the vector of category $A(B)$, the classifier is supposed to work properly as long as the value is positive (negative). This is why d is called as the "margin". d is usually set to 1 without loss of generality.

operation even when small deviation in $\vec{w} \cdot \vec{\xi}^{(j)}$ is caused by noise.

In order to facilitate our computation based on linear programming, let each variable w_i be decomposed into non-negative variables as follows

$$w_i = w_i^+ - w_i^-, \quad (i = 0, 1, 2, \dots, N)$$

where

$$w_i^+ \geq 0, w_i^- \geq 0 \quad (i = 0, 1, 2, \dots, N) \quad (2.5)$$

Thus (2.4) is rewritten as

$$\begin{aligned} \sum_{i=0}^N (w_i^+ - w_i^-) \xi_i^{(j)} &\geq d \quad \text{if } f = 1 \\ \sum_{i=0}^N (w_i^+ - w_i^-) \xi_i^{(j)} &\leq -d \quad \text{if } f = 0, \quad j=1, \dots, M, \end{aligned}$$

where $\xi_0^{(i)} = 1. \quad (2.6)$

If (2.6) (i.e.(2.4)) is consistent, there are generally an infinite number of solutions (\vec{w}, w_0) . Henceforth we will consider only a solution which minimizes a linear objective function of the weight vector

$$\vec{u} \cdot \vec{w}^+, \quad (2.7)$$

where \vec{u} is a $(2N + 2)$ - dimensional vector and

$$\vec{w}^+ = (w_0^+, w_0^-, w_1^+, \dots, w_N^-). \quad (2.8)$$

The coordinates \vec{u} will be determined in the following paragraph, depending upon which parameters of the classifier we want to minimize. The minimization of the objective function (2.7) under the constraints (2.5) and (2.6) is a typical linear programming problem.

The objective function is expressed in a general form in (2.7). However it can be correlated to the reliable operation of the classifier

as follows. Assume that the deviation of each input $\xi_i^{(j)}$ due to noise or other fluctuations in the circuit of the classifier is δ_i . Then the actual value of the left side of (2.4) is

$$\sum_{i=1}^N w_i (\xi_i^{(j)} + \delta_i) + w_0 = \sum_{i=1}^N w_i \xi_i^{(j)} + \sum_{i=1}^N w_i \delta_i + w_0.$$

Now let δ be the maximum of $|\delta_i|$ over all i ;

$$|\delta_i| \leq \delta, \quad i=1, 2, \dots, N.$$

Then we have

$$\left| \sum_{i=1}^N w_i \delta_i \right| \leq \delta \sum_{i=1}^N |w_i|.$$

Therefore if δ satisfies

$$\begin{aligned} \sum_{i=1}^N w_i \xi_i^{(j)} - \delta \sum_{i=1}^N |w_i| + w_0 &\geq 0 & \text{if } f = 1 \\ \sum_{i=1}^N w_i \xi_i^{(j)} + \delta \sum_{i=1}^N |w_i| + w_0 &\leq 0 & \text{if } f = 0, j = 1, \dots, M, \end{aligned} \quad (2.9)$$

the linear classifier operates correctly* even when all δ_i have the maximum deviation $+\delta$ or $-\delta$ (i.e. $|\delta_i| = \delta$ for all i). The maximum value of δ such that the classifier operates correctly is called the input tolerance. Denoting the input tolerance with γ , we have

$$\gamma = \min_j \frac{\left| \sum_{i=1}^N w_i \xi_i^{(j)} + w_0 \right|}{\sum_{i=1}^N |w_i|} \quad (2.10)$$

by (2.9). If we determine a solution (\vec{w}, w_0) such that γ is maximized,

* See the footnote on page 3.

the classifier is allowed to have the maximum deviation in its inputs and therefore we have maximized the reliability of the operation of the classifier. We will prove that the maximization of (2.10) is equivalent to the minimization of $\sum_{i=1}^N |w_i|$.

First of all, if \vec{w} is a weight vector which maximizes γ then $k \cdot \vec{w}$ also provides the same input tolerance, where k is any positive number such that $k \cdot \vec{w}$ is still a solution of (2.4). Now we can assume

$$\text{Min}_j \left| \sum_{i=1}^N w_i \xi_i^{(j)} + w_0 \right| = d \quad (2.11)$$

without loss of generality. If $\text{Min}_j \left| \sum_{i=1}^N w_i \xi_i^{(j)} + w_0 \right| = t > d$, we can simply multiply a certain positive number $k = \frac{d}{t} < 1$ in order to obtain (2.11) which of course still satisfies (2.4).

Consequently the maximization of γ is to minimize

$$\sum_{i=1}^N |w_i|$$

under conditions (2.4) and (2.11). In this case, however, we can delete condition (2.11). This is because if we minimize

$$\sum_{i=1}^N |w_i|$$

under condition (2.4) and obtain

$$\text{Min}_j \left| \sum_{i=1}^N w_i \xi_i^{(j)} + w_0 \right| = e > d,$$

then $\frac{d}{e} < 1$ and

$$\vec{w}' = \frac{d}{e} \cdot \vec{w}$$

satisfies (2.4). For this new weight vector \vec{w}'

$$\sum_{i=1}^N |w'_i| < \sum_{i=1}^N |w_i|$$

holds.

Thus a contradiction. Therefore we must have $e = d$, i.e. (2.11) is satisfied. An alternative proof is found in [5], [6].

By setting

$$u_0^+ = u_0^- = 0 \quad \text{and} \quad u_i^+ = u_i^- = 1 \quad \text{for } i = 1, 2, \dots, N,$$

the objective function (2.7) becomes

$$\sum_{i=1}^N (w_i^+ + w_i^-) \quad (2.12)$$

which represents $\sum_{i=1}^N |w_i|^*$. Solution of the linear program composed of

(2.12), (2.5) and (2.6) will lead to the design of a linear classifier with the maximum reliability of operation.

Another case was discussed in the papers [5], [6] when the values of $\xi_i^{(j)}$ are limited to +1 and -1, instead of real numbers. If

$w_i \xi_i^{(j)}$ ($i=0, 1, 2, \dots, N$) is permitted to deviate as much as $w_i \xi_i^{(j)} (1+\delta)$

or $w_i \xi_i^{(j)} (1-\delta)$ where δ is now a percentage deviation, then the input tolerance of a majority element is

$$\frac{1}{\bar{w}} \min_j \left| \sum_{i=0}^N w_i \xi_i^{(j)} \right|, \quad (2.13)$$

where

$$\bar{w} = \sum_{i=0}^N |w_i|.$$

* Obviously the minimization of (2.12) leads to the condition that

either w_i^+ or w_i^- is always 0. Then $\sum_{i=1}^N (w_i^+ + w_i^-) = \sum_{i=1}^N |w_i|$ follows.

And it was proved that the minimization of W is equivalent to the maximization of the input tolerance, i.e. the maximization of the reliability of operation. For this case we set $u_i^+ = u_i^- = 1$, $i = 0, 1, 2, \dots, N$ in (2.7).

Although the input tolerance is a typical objective to be optimized, if a certain parameter of the classifier can be represented in the form of (2.7), we can optimize other characteristic of the classifier rather than the input tolerance. In particular, when the integer programming is used, a wider variety of objective functions may be available for our choice.

So far we have assumed that the linear classifier processes the set of distinct patterns $\{\vec{\xi}^{(1)}, \dots, \vec{\xi}^{(M)}\}$. In other words, only $\vec{\xi}^{(1)}, \dots, \vec{\xi}^{(M)}$ are supplied repeatedly to the linear classifier as a time series $\dots, \vec{\xi}^{(t-1)}, \vec{\xi}^{(t)}, \vec{\xi}^{(t+1)}, \vec{\xi}^{(t+2)}, \dots$. The structure of the linear classifier was optimized for the set $\{\vec{\xi}^{(1)}, \dots, \vec{\xi}^{(M)}\}$.

Let us consider the case where the time series of pattern vectors gradually contain new pattern vectors for which the structure of the linear classifier is not optimum and instead some of the existing pattern vectors are eliminated from the time series. A few different schemes are conceivable to find whether the incoming pattern vector is considered as a new pattern vector to which the structure of the classifier is optimized.

These schemes will be discussed in Section 6. Here for the moment let us assume that the incoming pattern vector is found new by a certain scheme at some instant and that the linear classifier is supposed to process the new set of distinct pattern vectors $\{\vec{\xi}^{(2)}, \dots, \vec{\xi}^{(M+1)}\}$ instead of $\{\vec{\xi}^{(1)}, \dots, \vec{\xi}^{(M)}\}$. In other words, $\vec{\xi}^{(1)}$ is replaced by $\vec{\xi}^{(M+1)}$. Now the structure of the classifier is supposed to be optimized for $\{\vec{\xi}^{(2)}, \dots, \vec{\xi}^{(M+1)}\}$. The linear classifier should adapt itself to this new environment.

By changing the notations appropriately and multiplying the second inequality of (2.6) by -1, our linear program to minimize (2.12) under the constraints (2.5) and (2.6) is converted into

$$\begin{aligned} & \text{minimize } \vec{c} \vec{x} \\ & \text{subject to } A \vec{x} \geq \vec{b}^T \\ & \vec{x} \geq 0, \end{aligned} \tag{2.14}$$

where \vec{x} is an n-dimensional vector of unknown variables to be determined

and

$$\vec{c} = (c_1, \dots, c_n),$$

$$A = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix}, \quad (2.15)$$

$$\vec{b} = (b_1, \dots, b_m).$$

Note that A corresponds to the original given set of pattern vectors,

\vec{b} to d's, and \vec{x} to the original \vec{w} . \vec{c} represents the coefficients of the

objective function. Thus the adaptability problem of the classifier with the

change of environment may be stated as follows: Given an optimum solution for

the linear program (2.14), find an optimum solution for the new linear

program with the new coefficient row $\vec{a}^{(m+1)}$ replacing the oldest row

$\vec{a}^{(1)}$, representing the change from $\vec{\xi}^{(1)}$ to $\vec{\xi}^{(M+1)}$, i.e.,

$$\text{minimize} \quad \vec{c} \vec{x}$$

$$\begin{aligned} \text{subject to} \quad A' \vec{x} &\geq \vec{b}',^T \\ \vec{x} &\geq 0 \end{aligned} \quad (2.16)$$

where

$$A' = \begin{bmatrix} a_{21} & \dots & a_{2n} \\ \vdots & & \vdots \\ a_{m+1,1} & \dots & a_{m+1,n} \end{bmatrix}, \quad (2.17)$$

$$\vec{b}' = (b_2, \dots, b_{m+1}).$$

The effectiveness of the methods which we will discuss depends

on the validity of an assumption that an optimum solution of (2.16) does

not differ "too much" from the one of (2.14).

3. Outline of the Simplex Method

We assume that readers are familiar with the basic properties of linear inequalities and the simplex method, which is a computational procedure to solve a linear programming problem. However let us sketch important concepts which will be used often in the rest of this paper.

3.1 Duality Theorem of a Linear Program [7] [8] [9]

Consider the following linear program:

$$\begin{aligned} &\text{maximize} && \vec{b}^T \vec{v} \\ &\text{subject to} && A^T \vec{v} \leq \vec{c} \\ &&& \vec{v} \geq 0 \end{aligned} \tag{3.1.1}$$

where \vec{v} is an m -dimensional vector of unknown variables. This linear program is called the dual of (2.14). The coefficient matrix is the transpose of A in (2.14), and \vec{b} and \vec{c} are interchanged. It is known that when we solve either (3.1.1) or (2.14) by the simplex method, we will find either optimum solutions to both or infeasibility of the problems (one of them is possible unbounded.) Therefore we can solve the more convenient of (3.1.1) or (2.14).

3.2 Simplex Method.

Let us sketch the simplex method. See the literatures, [8] and [9], for details.

In this paper we will work on the dual problem (3.1.1) rather than the primal problem (2.14), because (3.1.1) in our case is more advantageous than (2.14) in several respects, as will be seen later.

We reformulate (3.1.1) in the following form by introducing so-called slack variables s_1, s_2, \dots, s_n :

maximize $\vec{b} \cdot \vec{v}$

subject to

$$n \left\{ \begin{array}{c} \overbrace{\left[\begin{array}{ccc} & & \\ & 1 & \dots & 0 \\ A^T & \vdots & & \vdots \\ & 0 & \dots & 1 \end{array} \right]}^{m+n} \begin{bmatrix} v_1 \\ \vdots \\ v_m \\ s_1 \\ \vdots \\ s_n \end{bmatrix} = \begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix} \end{array} \right. \quad (3.2.1)$$

$$\vec{v} \geq 0,$$

and

$$\vec{s} \geq 0.$$

The simplex method is a systematic procedure to choose a sequence of sets of n basic variables (only basic variables can assume non-zero values and non-basic variables are assigned zero.) out of the $m+n$ variables $v_1, \dots, v_m, s_1, \dots, s_n$ until we obtain an optimal solution consisting of basic variables which maximizes $\vec{b} \cdot \vec{v}$. A solution which satisfies constraints of (3.2.1) but is not necessarily optimal is called a feasible solution.

When $c_1, c_2, \dots, c_n \geq 0$, we may choose

$$v_1 = v_2 = \dots = v_m = 0$$

$$s_i = c_i \quad (i = 1, 2, \dots, n) \quad (3.2.2)$$

as an initial feasible solution. Let us assume that \vec{u} of (2.7) be non-negative as seen in (2.12) for example. Then, c_1, \dots, c_n are all non-negative as required in (3.2.2).

The simplex method is most conveniently described by using tableau representation. The first simplex tableau is shown in Fig. 3.2.1, whose elements are shown by the column vectors \vec{p} and \vec{q} , the row vector

\vec{r} , and the matrix H . Here

$\vec{q} = (q_1, \dots, q_n)^T$ is the values of basic variables in

the current tableau (There-

fore $\vec{q} \geq 0$ means that the

solution \vec{q} is feasible.) and

$\vec{p} = (p_1, \dots, p_n)^T$ denotes the

coefficients of the basic variables

\vec{q} in the objective function (i.e. b_j 's of \vec{b} in (3.2.1) corresponding

to the basic variables in \vec{q}). H may be divided into two parts

$$H = [\vec{h}_1, \vec{h}_2, \dots, \vec{h}_{m+n}] = [\underbrace{D}_m, \underbrace{B^{-1}}_n], \quad (3.2.3)$$

where \vec{h}_i for $1 \leq i \leq m$ is the column associated with the variable v_i and \vec{h}_i for $m+1 \leq i \leq m+n$ is the column associated with s_{i-m} . Each entry of $\vec{r} = (r_0, r_1, \dots, r_{m+n})$ can be obtained by the following:

$$r_0 = \vec{p} \cdot \vec{q}$$

$$r_i = \vec{p} \cdot \vec{h}_i - b_i \quad (i = 1, 2, \dots, m+n) \quad (3.2.4)$$

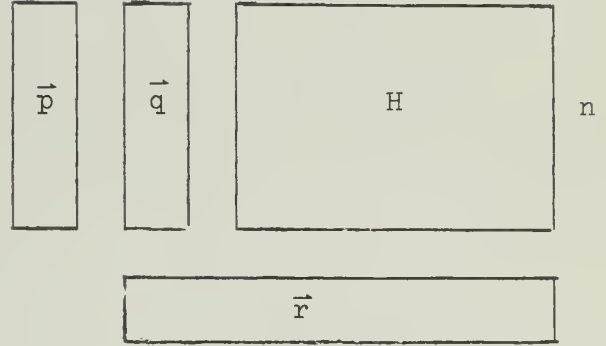
where b_{m+1}, \dots, b_{m+n} are set to 0 (see (3.2.1)). If we start with the

initial feasible solution (3.2.2), the initial tableau consists of

$$\left\{ \begin{array}{l} \vec{p} = \vec{0} \\ \vec{q} = \vec{c} \\ D = A^T \\ B^{-1} = I \quad (\text{the unit matrix}) \\ r_0 = 0 \\ r_i = -b_i \quad (i = 1, 2, \dots, m+n). \end{array} \right. \quad (3.2.5)$$

At each simplex tableau, there are three possibilities:

Fig. 3.2.1
Simplex Tableau
n + m



$$(1) \quad \vec{r} \geq 0.$$

$$(2) \quad \text{Existence of } i \text{ such that } r_i < 0 \text{ and } \vec{h}_i \leq 0.$$

$$(3) \quad \text{Neither (1) nor (2).}$$

Case (1) means that the feasible solution at the current tableau is optimal and the case (2) means that the linear program has an unbounded solution (i.e. the infeasibility of the primal problem (2.14) [7] [8] [9]). Whenever the case (3) is encountered, the so-called pivot operation is applied to the current tableau and the entries are transformed according to the simplex rule, deriving the next tableau. Then we examine which of the above three possibilities holds in the new tableau. With repeated derivation of tableaux we will eventually obtain the case (1) or (2). The pivot operation is essentially a process of replacing one basic variable (i.e. a column of the current tableau) by a new basic variable. See the literature [8] and [9] for the details of the derivation of simplex tableau.

If the simplex method ends up with the case (1), an optimal solution of (3.1.1) will be in column \vec{q} of the last tableau and an optimal solution of (2.14) will be $(r_{m+1}, \dots, r_{m+n})$ whose coordinates are the values of x_1, \dots, x_n , respectively.

Note that the following relations hold for every simplex tableau:

$$\begin{aligned} \vec{q} &= B^{-1} \vec{c} \\ \vec{h}_i &= B^{-1} \vec{a}_i^T \quad (i=1, 2, \dots, m+n) \end{aligned} \tag{3.2.6}$$

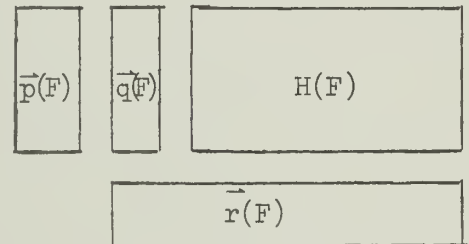
where \vec{a}_i^T is the i th column of $[A^T \ I]$ of (3.2.1). These relations will be used later.

4. Adaptation by the Simplex Method

When a classifier characterized by the linear program (2.14) adapts itself to a new environment with an old pattern vector replaced by a new one, it is characterized by the new linear program (2.16). Let us assume (for a while) that the linear programs (2.14) and (2.16) both are feasible. The case where they are infeasible will be considered at the end of this section and also in the next section.

Now suppose that we have solved the linear program (2.14) by using the dual formulation (3.1.1) and have obtained an optimum solution in the last simplex tableau as shown in Fig. 4.1. The adaptation is essentially an efficient method for deriving an initial tableau of the new problem (2.16) (strictly speaking, the dual formulation of (2.16)) and applying the pivot operations until a new optimal solution is obtained.*

Fig. 4.1
Last Simplex Tableau



* This type of problem has already been studied and is included in the subject of parametric programming. In this paper, however, we propose a new procedure rather than using the standard technique of parametric programming [8] [9], because (1) the pattern classification problem is suitable for the dual formulation with respect to signs of coefficients (no need of artificial variables), (2) subsequent procedure for adaptation is simpler and straightforward (the dual simplex method is not needed) and (3) our formulation is easily extendable to the integer linear programming formulation which will be discussed in Section 5.

This initial tableau is expected to yield faster convergence than the ordinary initial tableau given in (3.2.5).

In order to obtain this new initial tableau, we have to (1) eliminate the effect of the inequality

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n \geq b_1 \quad (4.1)$$

which corresponds to the pattern vector to be eliminated, and (2) introduce a new inequality

$$a_{m+1,1}x_1 + a_{m+1,2}x_2 + \dots + a_{m+1,n}x_n \geq b_{m+1} \quad (4.2)$$

which corresponds to a new pattern vector, without increasing the size of simplex tableau.

The first step corresponding to (1) is to set b_1 equal to $-S$ where S is a sufficiently large positive number such that (4.1) is satisfied for all feasible solutions of the remaining inequalities. This means that (4.1) is now non-restrictive. Let us recall that the current linear program is treated by the dual formulation and therefore the inequality (4.1) is associated with the first column $\vec{h}_1(F)$ of $H(F)$ in the last simplex tableau. The change of b_1 causes the change of an entry of $\vec{p}(F)$ which corresponds to a variable v_1 (or $\vec{h}_1(F)$) if v_1 is in the basis. But it causes no change if $\vec{h}_1(F)$ is not in the basis. The entries of $\vec{r}(F)$ are accordingly recalculated by (3.2.4). After this, simply delete the first column $\vec{h}_1(F)$ and r_1 . The deletion is permissible because $\vec{h}_1(F)$ henceforth will not enter the basis (since r_1 will not be negative.).

The second step corresponding to (2) above is to introduce the new inequality (4.2) into the column where (4.1) was eliminated. The new entries can be obtained by (3.2.4) and (3.2.6), i.e.

$$\vec{h}_1 = B^{-1} \vec{a}_{m+1}^T \quad (4.3)$$

where $\vec{a}_{m+1}^T = (a_{m+1,1}, a_{m+1,2}, \dots, a_{m+1,n})^T$

and

$$r_1 = \vec{p} \cdot \vec{h}_1 - b_{m+1}. \quad (4.4)$$

Note that \vec{q} was not changed in the above two steps and therefore the new tableau still has a feasible solution (although it may not be optimal). Thus this new tableau can be used as an initial tableau for the dual of the problem (2.16).

If all the entries of \vec{r} are non-negative, the old solution is optimal for the new problem also. But if \vec{r} contains some negative entries, apply the pivot operations until an optimal solution is derived for the new problem.

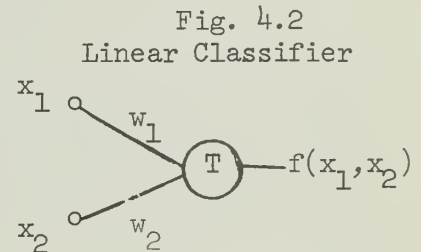
Although we discussed the procedure only for the case in which one inequality is replaced, extension to the case of more than one inequality is obvious.

The procedure would need less computation time if the coordinates of the new solution do not deviate "too much" from those of the old solution since the new pivot operation starts from a "closer" solution to the new problem than the ordinary initial solution of (3.2.5).

Example.

Let us pursue the adaptation of the linear classifier shown in Fig. 4.2, when pattern vectors change as follows from (4.5) through (4.7):

(i)	$(x_1, x_2):$	(11), (10)	yield $f = 1$	(4.5)
		(01)	yields $f = 0$	



$$\begin{array}{llll}
\text{(ii)} & (x_1, x_2): & (11), (00) & \text{yield } f = 1 \\
& & (01) & \text{yields } f = 0
\end{array} \tag{4.6}$$

$$\begin{array}{llll}
\text{(iii)} & (x_1, x_2): & (11) & \text{yields } f = 1 \\
& & (01), (00) & \text{yield } f = 0.
\end{array} \tag{4.7}$$

Corresponding to these sets of pattern vectors we have the following sets of inequalities by (2.4):

$$\begin{array}{ll}
\text{(i)} & \begin{array}{l} w_1 + w_2 + w_0 \geq 1^* \\ w_1 + w_0 \geq 1 \\ w_2 + w_0 \leq -1 \end{array}
\end{array} \tag{4.8}$$

$$\begin{array}{ll}
\text{(ii)} & \begin{array}{l} w_1 + w_2 + w_0 \geq 1 \\ w_0 \geq 1 \\ w_2 + w_0 \leq -1 \end{array}
\end{array} \tag{4.9}$$

$$\begin{array}{ll}
\text{(iii)} & \begin{array}{l} w_1 + w_2 + w_0 \geq 1 \\ w_0 \leq -1 \\ w_2 + w_0 \leq -1 \end{array}
\end{array} \tag{4.10}$$

Let us split these variables w_i 's as seen in (2.5) in order to keep the non-negativity of variables in the simplex method. The objective function to be minimized is now:

$$|w_1| + |w_2| = w_1^+ + w_1^- + w_2^+ + w_2^-. \tag{4.11}$$

Renaming these split variables and changing the direction of some inequalities, the above sets of inequalities may be rewritten as follows, corresponding to (2.14):

* d in (2.4) is set to 1 for simplicity.

$$\begin{aligned}
& x_1 - x_2 + x_3 - x_4 + x_5 - x_6 \geq 1 & v_1 \\
(i) \quad & x_1 - x_2 & + x_5 - x_6 \geq 1 & v_2 \\
& & - x_3 + x_4 - x_5 + x_6 \geq 1 & v_3
\end{aligned} \tag{4.12}$$

$$\begin{aligned}
& x_1 - x_2 + x_3 - x_4 + x_5 - x_6 \geq 1 & v_1 \\
(ii) \quad & & x_5 - x_6 \geq 1 & v_4 \\
& & - x_3 + x_4 - x_5 + x_6 \geq 1 & v_3
\end{aligned} \tag{4.13}$$

$$\begin{aligned}
& x_1 - x_2 + x_3 - x_4 + x_5 - x_6 \geq 1 & v_1 \\
(iii) \quad & & - x_5 + x_6 \geq 1 & v_5 \\
& & - x_3 + x_4 - x_5 + x_6 \geq 1 & v_3
\end{aligned} \tag{4.14}$$

where v_i is shown simply for identification of each inequality. The objective function in the renamed variables is

$$x_1 + x_2 + x_3 + x_4 \tag{4.15}$$

Note that the following relations hold among the original and renamed variables

$$\begin{aligned}
x_1 - x_2 &= w_1^+ - w_1^- = w_1 \\
x_3 - x_4 &= w_2^+ - w_2^- = w_2 \\
x_5 - x_6 &= w_0^+ - w_0^- = w_0 .
\end{aligned} \tag{4.16}$$

Assume that our classifier has the first set of pattern vectors (i). An initial tableau for the dual of this linear program will be derived by (3.2.5). Table 4.1 shows the initial tableau derived. After applying the pivot operation three times (Tables 4.2 and 4.3 and 4.4), an optimal solution will result since Table 4.4 contains no negative entry in \bar{r} . The solution is obtained in (r_4, \dots, r_9) , i.e.,

$$\begin{aligned}
 x_1 &= 2 \\
 x_6 &= 1 \\
 x_2 &= x_3 = x_4 = x_5 = 0 \quad ,
 \end{aligned}
 \tag{4.17}$$

which implies

$$\begin{aligned}
 w_1 &= 2 \\
 w_2 &= 0 \\
 w_0 &= -1 \quad .
 \end{aligned}
 \tag{4.18}$$

Then assume that the first set of pattern vectors is changed to the second set (ii). According to the procedure discussed, eliminate the inequality which corresponds to the old pattern vector to be replaced,

$$x_1 - x_2 + x_5 - x_6 \geq 1 \quad v_2 \tag{4.19}$$

and instead introduce the inequality

$$x_5 - x_6 \geq 1 \quad v_4 \quad . \tag{4.20}$$

Variables in the basis	\vec{p} ↓	\vec{b}	1	1	1	0	0	0	0	0	0	column name
		\vec{q} ↓	v_1	v_2	v_3	s_1	s_2	s_3	s_4	s_5	s_6	
s_1	0	1	1	1	0	1	0	0	0	0	0	
s_2	0	1	-1	-1	0	0	1	0	0	0	0	
s_3	0	1	1	0	-1	0	0	1	0	0	0	
s_4	0	1	-1	0	1	0	0	0	1	0	0	
s_5	0	0	1	1	-1	0	0	0	0	1	0	
s_6	0	0	-1	-1	1	0	0	0	0	0	1	
	\vec{r}	0	-1	-1	-1	0	0	0	0	0	0	

Table 4.1 Initial Tableau for the First
Set of Pattern Vectors

			v_1	v_2	v_3							
s_1	0	1	0	0	1	1	0	0	0	-1	0	
s_2	0	1	0	0	-1	0	1	0	0	1	0	
s_3	0	1	0	-1	0	0	0	-1	0	-1	0	
s_4	0	1	0	1	0	0	0	0	1	1	0	
v_1	1	0	1	1	-1	0	0	0	0	1	0	
s_6	0	0	0	0	0	0	0	0	0	1	1	
		0	0	0	-2	0	0	0	0	1	0	

Table 4.2 Intermediate Tableau for the First
Set of Pattern Vectors

v_3	1	1	0	0	1	1	0	0	0	-1	0
s_2	0	2	0	0	0	1	1	0	0	0	0
s_3	0	1	0	-1	0	0	0	1	0	-1	0
s_4	0	1	0	1	0	0	0	0	1	1	0
v_1	1	1	1	1	0	1	0	0	0	0	0
s_6	0	0	0	0	0	0	0	0	0	(1)	1
		2	0	0	0	2	0	0	0	-1	0

Table 4.3
Intermediate Tableau for the First
Set of Pattern Vectors

v_3	1	1	0	0	1	1	0	0	0	0	-1
s_2	0	2	0	0	0	1	1	0	0	0	0
s_3	0	1	0	-1	0	0	0	1	0	0	1
s_4	0	1	0	1	0	0	0	0	1	0	-1
v_1	1	1	1	1	0	1	0	0	0	0	0
s_5	0	0	0	0	0	0	0	0	0	1	1
		2	0	0	0	2	0	0	0	0	1

Table 4.4
Optimal Tableau for the First Set of
Pattern Vectors

		\vec{b}	1	-100	1	0	0	0	0	0	
v_3	1	1	0	0	1	1	0	0	0	0	-1
s_2	0	2	0	0	0	1	1	0	0	0	0
s_3	0	1	0	-1	0	0	0	1	0	0	1
s_4	0	1	0	1	0	0	0	0	1	0	-1
v_1	1	1	1	1	0	1	0	0	0	0	0
s_5	0	0	0	0	0	0	0	0	0	1	1
		2	0	101	0	2	0	0	0	0	1

Table 4.5
Elimination of the Old Vector V_2

Replace 1 in b_2 (of v_2) by a sufficiently small number, say - 100.

Since v_2 is not in the basis in Table 4.4, the replacement does not cause any change of any entry except r_2 of Table 4.4. Table 4.5 shows the resultant tableau. Next delete h_2 and r_2 from Table 4.5 and fill in new entries which are derived from the new inequality (4.20) according to (4.3) and (4.4). Note that B^{-1} is $[h_4, h_5, \dots, h_9]$ in this case. The result is shown in Table 4.6. Since there is a negative entry in r , apply the pivot operation. After two applications, a new optimal solution is derived as shown in Table 4.7. It is

$$x_1 = 2, x_4 = 2, x_5 = 1 \quad (4.21)$$

$$x_2 = x_3 = x_6 = 0$$

which lead to

$$w_1 = 2$$

$$w_2 = -2 \quad (4.22)$$

$$w_0 = 1 .$$

			v_1	v_4	v_3	s_1	s_2	s_3	s_4	s_5	s_6
v_3	1	1	0	-1	1	1	0	0	0	0	1
s_2	0	2	0	0	0	1	1	0	0	0	0
s_3	0	1	0	-1	0	0	0	1	0	0	1
s_4	0	1	0	1	0	0	0	0	1	0	-1
v_1	1	1	1	0	0	1	0	0	0	0	0
s_5	0	0	0	0	0	0	0	0	0	1	1
		2	0	-2	0	2	0	0	0	0	1

Table 4.6

Initial Tableau for the Second

Set of Pattern Vectors

	\vec{p} ↓	\vec{q} ↓									
v_3	1	2	0	0	1	1	0	0	1	0	0
s_2	0	2	0	0	0	1	1	0	0	0	0
s_3	0	2	0	0	0	0	0	1	1	0	0
v_4	1	1	0	1	0	0	0	0	1	1	0
v_1	1	1	1	0	0	1	0	0	0	0	0
s_6	0	0	0	0	0	0	0	0	0	1	1
		4	0	0	0	2	0	0	2	1	0

Table 4.7

Optimal Tableau for the Second

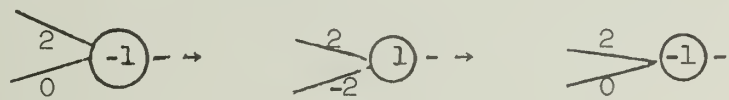
Set of Pattern Vectors

The change from the second set of pattern vectors (ii) to the third set (iii) can be treated in a similar manner. In this case \bar{p} column is changed because v_4 is in the basis as seen in Table 4.8. Then Table 4.9 is the initial tableau for the third set (iii). An optimal solution is obtained in Table 4.10 after two pivot operations. It is

$$w_1 = 2, w_2 = 0, w_0 = -1 \tag{4.23}$$

From (4.18), (4.22), (4.23), we can see that the structure of the linear classifier has changed as shown in Fig. 4.3.

Fig. 4.3 Adaptation of Linear Classifier



			v_1	v_4	v_3	s_1	s_2	s_3	s_3	s_4	s_5
v_3	1	2	0	0	1	1	0	0	1	0	0
s_2	0	2	0	0	0	1	1	0	0	0	0
s_3	0	2	0	0	0	0	0	1	1	0	0
v_4	-100	1	0	1	0	0	0	0	1	1	0
v_1	1	1	1	0	0	1	0	0	0	0	0
s_6	0	0	0	0	0	0	0	0	0	1	1
.		-97	0	0	0	2	0	0	-99	-100	0

Table 4.8

Elimination of the Old Vector v_4

			v_1	v_5	v_3	s_1	s_2	s_3	s_4	s_5	s_6
v_3	1	2	0	0	1	1	0	0	1	0	0
s_2	0	2	0	0	0	1	1	0	0	0	0
s_3	0	2	0	0	0	0	0	1	1	0	0
v_4	-100	1	0	-1	0	0	0	0	1	1	0
v_1	1	1	1	0	0	1	0	0	0	0	0
s_6	0	0	0	0	0	0	0	0	0	1	1
		-97	0	99	0	2	0	0	-99	-100	0

Table 4.9

Initial Tableau for the Third

Set of Pattern Vectors

v_3	1	1	0	1	1	1	0	0	0	0	1
s_2	0	2	0	0	0	1	1	0	0	0	0
s_3	0	1	0	1	0	0	0	1	0	0	1
s_4	0	1	0	-1	0	0	0	0	1	0	-1
v_1	1	1	1	0	0	1	0	0	0	0	0
s_5	0	0	0	0	0	0	0	0	0	1	1
		2	0	0	0	2	0	0	0	0	1

Table 4.10

Optimal Tableau for the Third

Set of Pattern Vectors

Now let us consider the case in which the given set of pattern vectors are not separable, i.e., (2.14) (or (2.16)) is not feasible. By adding an artificial variable to each inequality of (2.14), (2.14) can be rewritten as

$$\begin{aligned} a_{i1} x_1 + \dots + a_{in} x_n + t_i &\geq b_i \\ t_i &\geq 0 & (i = 1, 2, \dots, m) \\ x_j &\geq 0 & (j = 1, 2, \dots, n). \end{aligned} \quad (4.24)$$

Obviously (4.24) is feasible because the set of inequalities are satisfied by a sufficiently large positive value of each t_i . However the positiveness of t_i does not mean that the corresponding i th inequality of (2.14) is satisfied. In order to circumvent this fact let us use the following new objective function.

$$V \sum_{i=1}^m t_i + \vec{c} \cdot \vec{x} \quad (4.25)$$

where V is some large positive number. Roughly speaking, the minimization of this objective function means that it first minimizes $\sum_{i=1}^m t_i$ which will tend to minimize the number of incorrectly separated vectors and that it next minimizes $\vec{c} \cdot \vec{x}$, the original objective function.* The set of inequalities (4.24) with the objective function (4.25) can be solved in a similar manner as the case of (2.14), since (4.24) is now feasible. Although the number of artificial variables increases

* Ideally, $\sum_{i=1}^m t_i$ is to be minimized, no matter what value $\vec{c} \cdot \vec{x}$ assumes. However, this is not achieved by (4.25) because t_i 's are continuous variables. In other words, the value of $\sum_{i=1}^m t_i$, when (4.25) is minimized, depends upon the relative size of V and $\vec{c} \cdot \vec{x}$ and may not be minimized. Note that if t_i 's are discrete variables, $\sum_{i=1}^m t_i$ is minimized as will be discussed in Section 5.

indefinitely if the adaptation is repeated infinitely, the storage space increase can be avoided by replacing the old artificial variable to be eliminated by the new artificial variable.

The objective function (4.25), however, does not always lead to the minimum number of erroneously separated pattern vectors. As will be discussed in the next section, the exact minimization of the number of incorrectly separated vectors will be obtained by using integer linear programming.

5. Formulation and Adaptation by Integer Linear Programming

Given a non-separable set of pattern vectors, one realization of a linear classifier is to minimize the number of incorrectly separated vectors. This can be achieved by the following integer linear programming approach.

Corresponding to each input vector $\vec{\xi}^{(j)}$ of (2.4),

$$\vec{w} \cdot \vec{\xi}^{(j)} + w_0 \geq d \quad \text{if } f(\vec{\xi}^{(j)}) = 1 \quad (5.1A)$$

or

$$-\vec{w} \cdot \vec{\xi}^{(j)} - w_0 \geq d \quad \text{if } f(\vec{\xi}^{(j)}) = 0, \quad (5.1B)$$

formulate the two inequalities for each of (5.1A) and (5.1B) as follows:

$$\vec{\xi}^{(j)} \vec{w} + w_0 + UP_j \geq d \quad (5.2A)$$

$$\vec{\xi}^{(j)} \vec{w} + w_0 - U(1 - P_j) \leq -d \quad \text{if } f(\vec{\xi}^{(j)}) = 1 \quad (5.3A)$$

or

$$-\vec{\xi}^{(j)} \vec{w} - w_0 + UP_j \geq d \quad (5.2B)$$

$$-\vec{\xi}^{(j)} \vec{w} - w_0 - U(1 - P_j) \leq -d \quad \text{if } f(\vec{\xi}^{(j)}) = 0 \quad (5.3B)$$

where P_j is a variable which assumes 1 or 0, and where U is a sufficiently large positive number to insure the following property.

When $P_j = 0$, (5.2A and B) is obviously identical to (5.1A and B) respectively and (5.3A and B) is satisfied by any value of $\vec{\xi}_i^{(j)}$. This means (5.1A or B) is satisfied by \vec{w} and accordingly the pattern vector is classified correctly. On the other hand, when $P_j = 1$, (5.3A and B) become

$$\vec{w} \cdot \vec{\xi}^{(j)} + w_0 \leq -d \quad \text{if } f = 1 \quad (5.4A)$$

or

$$-\vec{w} \cdot \vec{\xi}^{(j)} - w_0 \leq -d \quad \text{if } f = 0. \quad (5.4B)$$

The pattern vector is separated incorrectly. Therefore

$$\sum_{j=1}^m P_j \quad (5.5)$$

shows the number of incorrectly separated pattern vectors.

Now consider the objective function:

$$\text{minimize} \quad V \sum_{j=1}^m P_j + \vec{u} \vec{w}^+, \quad (5.6)^*$$

where \vec{u} is the vector in (2.7) and where V is a sufficiently large positive number (i.e. $V > \text{Max } \vec{u} \vec{w}^+$). Different from the objective function (4.25) in Section 4, the minimization of this objective function means the minimization of both $\sum P_j$ and $\vec{u} \vec{w}^+$, because $\sum P_j$ assumes discrete values only. This is the property which we desired.

Of course, we can consider other objective functions if they are needed. The incorporation of integer variables extensively widens the concept of objective functions. For example, it is possible to minimize the number of non-zero weights, i.e., the number of inputs actually needed, by reformulating the integer linear programming with additional integer variables and changing the objective function.**

This new linear program is a mixed-integer linear programming problem because P_j must be integral. There are several known methods to solve this type of problem. Among them, we will use Gomory's all-integer integer linear programming method^[10] by assuming that the other variables are also integers, without loss of generality in the

* A weighted sum $\sum e_j P_j$ may be used, when there is preference among pattern vectors.

** This formulation is due to F. Chen.

sense that both formulations will yield the same minimum number of incorrectly separated pattern vectors. This integral condition is adopted because his method is very similar to the simplex method which was applied to the dual problem discussed in the earlier sections. The adaptation process will be accordingly analogous to the case of these earlier sections.

In order to apply Gomory's method we need another constraints,

$$P_j \leq 1, \quad j = 1, 2, \dots, m. \quad (5.7)$$

For notational convenience, let us henceforth denote our new problem of (5.2), (5.3), (5.7) together with the objective function (5.6) by:

$$\begin{aligned} &\text{minimize} && \vec{c}^* \vec{w} \\ &\text{subject to} && \\ &&& A^* \vec{x} \geq \vec{b}^* T \\ &&& \vec{x} \geq 0 \text{ and integers,} \end{aligned} \quad (5.8)$$

where \vec{x} is an n' -dimensional vector of unknown variables to be determined and

$$\begin{aligned} \vec{c}^* &= (c_1^*, \dots, c_n^*) \\ A^* &= \begin{bmatrix} a_{11}^*, \dots, a_{1n}^* \\ \vdots \\ a_{m'1}^*, \dots, a_{m'n}^* \end{bmatrix} \\ \vec{b}^* &= (b_1^*, \dots, b_{m'}^*). \end{aligned} \quad (5.9)$$

In our case, m' is equal to $3M$ and n' is the number of variables $x_1, \dots, x_n, P_1, \dots, P_m$, which is $2(N+1) + 3M$.

Before discussing our adaptation process, let us outline Gomory's method. Rigorous description and proofs are found in [10].

The method is very similar to the ordinary simplex method applied to the dual formulation. A simplex tableau of the method is shown in Fig. 5.1. Each column of H corresponds to an inequality of (5.8). (Slack variables are taken into the account. See (3.2.1).)

However when some entries of the row \vec{r} are negative,

Fig. 5.1 Tableau for

Integer Linear Programming

we pick a certain column

according to a rule

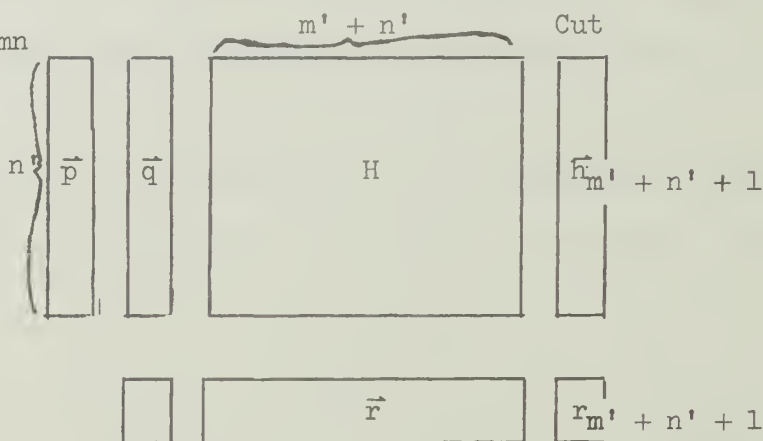
stated in [10], and

form a new column

called a "cut" from

that column. The

cut is the column



$(\vec{h}_{m' + n' + 1}, \vec{r}_{m' + n' + 1})$ in Fig. 5.1. The procedure to derive the cut is also discussed in [10]. Then the ordinary pivot operation is performed using this cut. This process is repeated until we obtain all non-negative entries in \vec{r} or find at least one column, say i , such that $r_i < 0$ and $\vec{h}_i \leq 0$ (infeasibility of (5.8)). Thus, only cuts can enter the basis. The initial tableau could be the one as shown in (3.2.5).

Although Gomory did not include the column \vec{p} in his method, \vec{p} is necessary for our adaptation procedure. In the simplex method of Section 3, the column \vec{p} had the following meaning: suppose p_i corresponds to column \vec{h}_j which was transformed from the inequality

$$a_{j1} x_1 + \dots + a_{jn} x_n \geq b_j,$$

and then $p_i = b_j$ holds.

The column \vec{p} in Fig- 5.1 of integer linear programming also has the similar meaning except that since each basis is derived through the cut rather than each inequality as in the simplex method, p_i corresponds to the right hand side of the cut through which p_i was introduced.

\vec{p} can be calculated by (3.2.4) each time a cut is introduced into a basis. Assume that a cut is to be introduced into the i -th row by the pivot operation, then the equation from (3.2.4)

$$r_{m' + n' + 1} = \vec{p} \cdot \vec{h}_{m' + n' + 1} - b_{m' + n' + 1}^*$$

is rewritten as

$$b_{m' + n' + 1}^* = \vec{p} \cdot \vec{h}_{m' + n' + 1} - r_{m' + n' + 1} \quad (5.10)$$

where only $b_{m' + n' + 1}^*$ is unknown variable because all the inequality in the $m' + n' + 1$ -th column is obtained by Gomory's algorithm [10]. Now the new p_i which will be used in the next step is

$$p_i = b_{m' + n' + 1}^* \quad (5.11)$$

The other entries of \vec{p} do not change.

Applying this pivot operation until all the entries in \vec{r} become non-negative, an optimal integer solution for (5.8) is found in $(r_{m' + 1}, \dots, r_{m' + n'})$ as in the case of the ordinary simplex method.

Suppose that we have obtained an optimal solution for the problem (5.8). Now eliminate the effect of the oldest pattern vector and let us introduce a new pattern vector. Let us express the oldest pattern vector by

$$a_{11} x_1 + \dots + a_{1n} x_n + UP_1 \geq b_1 \quad (5.12)$$

$$a_{11} x_1 + \dots + a_{1n} x_n - U(1-P_1) \leq b_1 - 1 \quad (5.13)$$

or by renaming the variable coefficients and the constant and by changing the direction of the second inequality,

$$a_{11}^* x_1 \dots + a_{1n'}^* x_{n'} \geq b_1^* \quad (5.14)$$

$$a_{21}^* x_1 + \dots + a_{2n'}^* x_{n'} \geq b_2^* . \quad (5.15)$$

Similarly let us denote the new pattern vector by

$$a_{m'+1,1}^* x_1 + \dots + a_{m'+1,n'}^* x_{n'} \geq b_{m'+1}^* \quad (5.16)$$

$$a_{m'+2,1}^* x_1 + \dots + a_{m'+2,n'}^* x_{n'} \geq b_{m'+2}^* . \quad (5.17)$$

which correspond to (5.2 or 5.3) for the given pattern vector.

The elimination of the oldest pattern vector can be done in a similar way as before: replace b_1^* and b_2^* by $-S$ where S is a sufficiently large positive number. However, this affects the tableau only through the cuts which are derived from the inequalities, (5.14) and (5.15). (The classifier must have memory space for this information.) In other words, this means replacement of the entries of \vec{p} which correspond to the cuts which were derived from (5.14) (5.15), by $-S$ so that the cuts become non-restrictive. More than one entry or no entry may exist, depending on what the current basis is.

The next step is to delete the columns \vec{h}_1 , \vec{h}_2 and the entries r_1 , r_2 which correspond to (5.14) and (5.15). And then calculate the new columns \vec{h}_1 , \vec{h}_2 from the new inequalities, (5.16) and (5.17), by using the relation (3.2.6). Of course, $B^{-1} = [\vec{h}_{m'+1}, \dots, \vec{h}_{m'+n'}]$. Note that the variable $P_{m'+1}$ which is implicitly in (5.16) and (5.17) should be replaced by P_1 of the oldest pattern vector in order to prevent the increase of the number of variables. Finally the new row \vec{r} can be obtained by (3.2.4). (Here we need \vec{p} which was obtained by (5.11).)

If there are negative entries in the new row \vec{r} , Gomory's pivot operation is repeated as discussed before until we obtain an optimal solution. This completes our adaptation procedure. (Note that the condition $r_i \leq 0$ and $\bar{h}_i \leq 0$ will not be reached since our problem is always feasible.) The whole process will be repeated when pattern vectors are changed.

6. Entire Scheme Of Adaptation And New Pattern Vector Identification

The adaptation procedure of a linear classifier which we have discussed in the previous sections is summarized as follows: Given an optimal structure for the set of distinct pattern vectors $\{\vec{\xi}^{(1)}, \vec{\xi}^{(2)}, \dots, \vec{\xi}^{(M)}\}$, the classifier readjusts itself so that its structure is optimal for the new set of distinct pattern vectors $\{\vec{\xi}^{(2)}, \dots, \vec{\xi}^{(M)}, \vec{\xi}^{(M+1)}\}$ where $\vec{\xi}^{(1)}$ is replaced by $\vec{\xi}^{(M+1)}$. However we did not discuss how to identify $\vec{\xi}^{(M+1)}$ among the pattern vectors incoming as a time series, in order to get the new set $\{\vec{\xi}^{(2)}, \dots, \vec{\xi}^{(M+1)}\}$ for which the classifier's structure ought to be optimal. There are a few different identification schemes conceivable. These schemes will be discussed in this section.

The entire system of the linear classifier is illustrated in Fig.6.1. Block C stores the last simplex tableau for $\{\vec{\xi}^{(1)}, \dots, \vec{\xi}^{(M)}\}$. Block A examines an incoming vector $\vec{\xi}'$ and decides whether it should be considered as a new pattern vector $\vec{\xi}^{(M+1)}$ by checking the information about $\{\vec{\xi}^{(1)}, \dots, \vec{\xi}^{(M)}\}$ stored in Block C (the adjustment of structure of the classifier results), or not (no adjustment results). If the new pattern vector $\vec{\xi}^{(M+1)}$ is identified, Block B computes the new optimal structure for $\{\vec{\xi}^{(2)}, \dots, \vec{\xi}^{(M)}, \vec{\xi}^{(M+1)}\}$ by starting from the last simplex tableau for $\{\vec{\xi}^{(1)}, \dots, \vec{\xi}^{(M)}\}$ stored in Block C. Using this new structure, Block D classifies the incoming vector $\vec{\xi}'$. (The $\vec{\xi}'$ must be kept in a buffer memory during the identification process by Block A and the computation of the new structure by block B.) The last simplex tableau is stored in Block C. Note that the last simplex tableau now stores the information about $\{\vec{\xi}^{(2)}, \dots, \vec{\xi}^{(M+1)}\}$ instead of $\{\vec{\xi}^{(1)}, \dots, \vec{\xi}^{(M)}\}$. This completes a cycle of adaptation procedure.

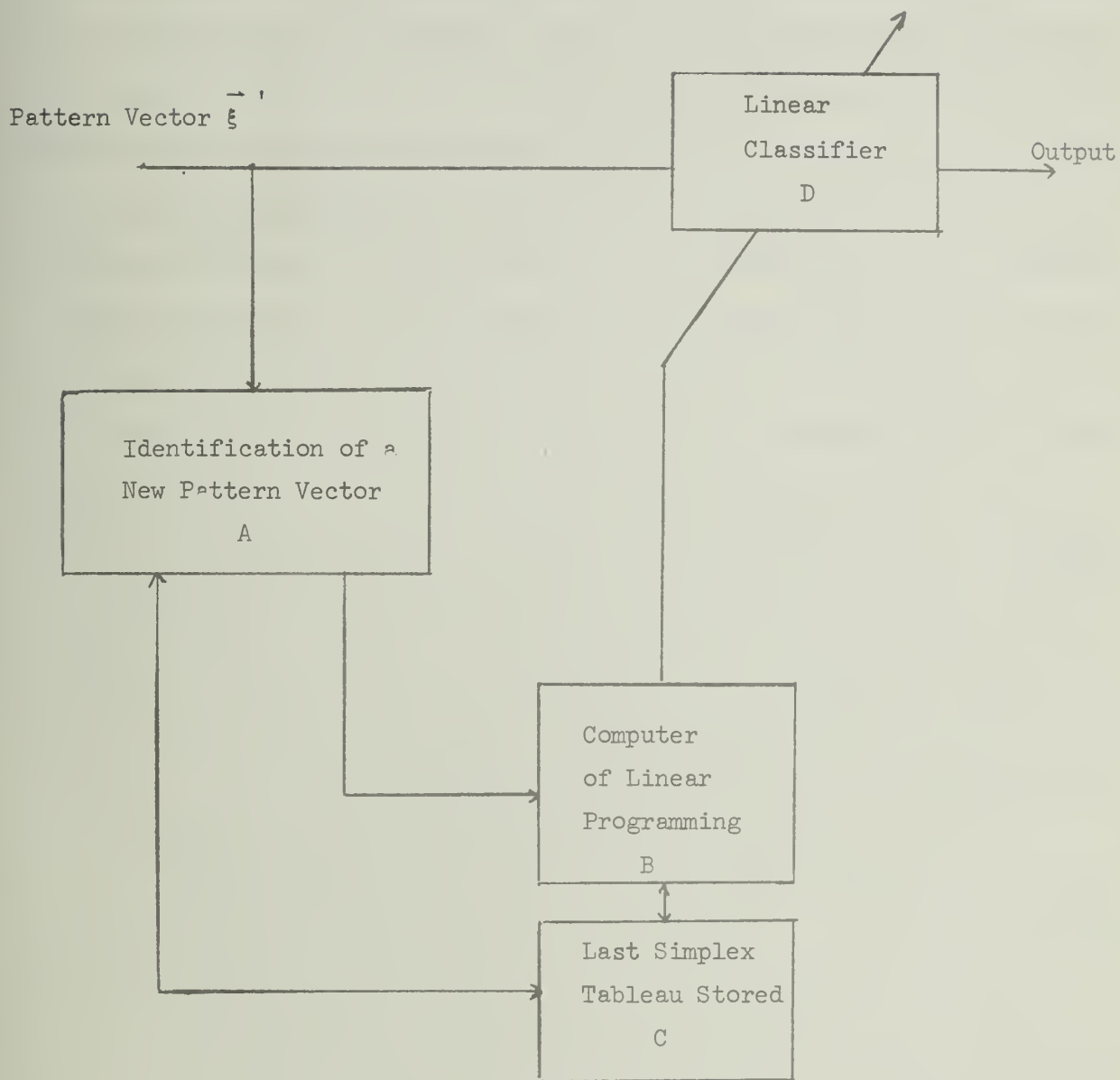


Fig. 6. 1. Adaptation Scheme

The entire procedure is characterized by specifying how Block A identifies the new pattern vector $\vec{\xi}^{(M+1)}$ which is then processed by Block B. If Block A identifies very often an incoming vector $\vec{\xi}$ as a new pattern vector $\vec{\xi}^{(M+1)}$, then the whole adaptation procedure also must work as often and it slows down the processing speed of the classifier. In this case, however, the accuracy of classification by the classifier is maintained. On the other hand, if very few incoming pattern vectors are processed for adaptation, the processing speed will be much faster. (Since the interval between adjacent adaptations is long, the buffer memory will not be filled up by incoming pattern vectors which are waiting for the classification. Therefore the transmission rate of pattern vectors can be speeded up).

When memory space of Block C is limited, the size of M is limited. Therefore if the number of distinct vectors in the time series is more than M, some selection of M vectors out of all distinct vectors must be made for adaptation. Therefore, generally speaking, even if the time series contains new pattern vectors the adaptation may not take place and the classifier may not work correctly for each pattern vector.

In the following, three simple schemes to identify new pattern vectors are arranged in the descending order of adaptation frequency.

Adaptation scheme (1); Regard every incoming pattern vector as the new vector $\vec{\xi}^{(M+1)}$, no matter whether it is actually new or not. This eliminates the checking procedure of Block A. This scheme guarantees that the current structure is optimal for the last M pattern vectors. But the classifier does adaptation all the time and the optimality of the structure is valid only over the last M pattern vectors.

Adaptation scheme (2); Block A checks whether each incoming pattern vector is new or not, by comparing it with those stored in Block C. If it is new, the classifier solves a linear program, otherwise it does not. This scheme is different from (1) in that the structure in (2) is optimal for M distinct pattern vectors which appeared most recently. These M distinct pattern vectors are, of course, stored in the simplex tableau.

Adaptation scheme (3); This scheme is very similar to the so-called error-correction procedure [1] [2] of an adaptive element, as far as the selection of a pattern vector is concerned. $\vec{\xi}'$ is regarded as the new pattern vector $\vec{\xi}^{(M+1)}$, only if $\vec{\xi}'$ is classified erroneously by the current structure. The checking of whether it is classified correctly or not is done by simply substituting the current solution to the corresponding inequality. Note that if the pattern $\vec{\xi}'$ is separated correctly, the current optimum structure for $\{\vec{\xi}^{(1)}, \dots, \vec{\xi}^{(M)}\}$ is also optimum for $\vec{\xi}'$ in the sense that it is optimum for $\{\vec{\xi}^{(1)}, \dots, \vec{\xi}^{(M)}, \vec{\xi}'\}$. However, obviously the optimum structure for $\{\vec{\xi}^{(2)}, \dots, \vec{\xi}^{(M)}, \vec{\xi}'\}$ which is obtained by replacing $\vec{\xi}^{(1)}$ by $\vec{\xi}'$, may be different from the current structure.

As a result the adaptation will take place less frequently than the other schemes. In other words, majority of incoming vectors are not included in the M vectors but it is likely that they are correctly classified when they appear again. The current structure, however, is optimum only for the last M pattern vectors for which the classifier solved linear programs, because the readjusted structure may be no longer optimum for the pattern vectors which were not identified as new pattern vectors and therefore have nothing to do with the current structure.

In addition to the above three schemes, depending upon the given situation, various sophisticated approaches might be also conceivable including the addition of the random sampling approach. However, various aspects of incoming pattern vectors should be examined and possibly experimental justification is necessary, in order to find a proper scheme .

7. Computational Experiment of Adaptation by the Simplex Method

The adaptation algorithm by the simplex method discussed in Section 4 was tested by actually solving a sequence of sets of pattern vectors. Each pattern set consisted of 30 30-dimensional pattern vectors whose coordinates are 1 or 0. Each two consecutive pattern sets differ in exactly one pattern vector so that the adaptation algorithm may be applied. First, we generated 60 pattern vectors together with categories to which pattern vectors should belong, by using random number generator which produces 1 and 0 with the equal probability. Second, the i -th pattern set $S(i)$ was defined as follows:

$$S(i) = \{ \vec{\xi}(i), \dots, \vec{\xi}(i + 29) \}, i = 1, 2, \dots, 31, \text{ where}$$

 $\vec{\xi}(j), j = 1, 2, \dots, 60, \text{ is the } j\text{-th generated pattern vector. These}$
31 problems corresponding to $S(1), \dots, S(31)$ were transformed into the sets of linear inequalities as illustrated in the example in Section 4, and then solved on the IBM 360/75 computer by using IBM Mathematical Programming System. (Although MPS cannot perform exactly the same procedure as described in Section 4, an equivalent modified test was tried in order to observe the number of necessary pivot operations.)

The result is that when we solve the 31 problems separately, the average number of pivot operations is 51.3 for each problem whereas 18.5 pivot operations are needed in our case of the adaptation scheme. The saving of pivot operations was about two third. This may be encouraging because the tested problem seems difficult for our approach since pattern vectors which are coming in or going out are generated independently of other pattern vectors in the set (i.e., by the random number generator) and accordingly the new solution may not have much relationship with the old one.

As is expected, the number of pivot operations fluctuates more widely in our adaptation scheme than the case of solving 31 problems separately. This is because the adaptation procedure usually needs more pivot operations if the pattern vector, which leaves the current pattern set, is in the basis of the last simplex tableau, than if it is not in the basis.

8. Conclusion

We have discussed the adaptation procedure of a linear classifier based on a linear programming method, making use of the easy incorporation of a new constraint in the simplex method for the dual formulation. Its advantage over the existing ordinary adaptive procedures (such as the error-correction method [1] [2]) is that the optimality of a parameter of the classifier, such as the input tolerance which is a measure of the reliability of the classifier operation, is guaranteed.

Even when a given set of pattern vectors is not linearly separable, the adaptation procedure based on integer linear programming attains the minimality of the number of incorrectly separated pattern vectors as well as the input tolerance of the classifier.

On the other hand, the disadvantage of the approach is the need of somewhat complicated computation for each adaptation, and the need of storage space for the simplex tableau.

The computational experiment in Section 7 is encouraging and may indicate the feasibility of this kind of system.

Acknowledgment

The authors would like to appreciate C. R. Baugh for his assistance in performing the experiment of Section 7 and for valuable comments on the manuscript.

References

- (1) N. J. Nilsson Learning Machines, McGraw-Hill, 1963, New York.
- (2) F. Rosenblatt, Principle of Neurodynamics, Spartan Books, Washington D. C., 1961
- (3) O. L. Mangasarian, "Linear and Nonlinear Separation of Patterns by Linear Programming", Operations Research, vol. 13, No. 3, pp. 444-452, May-June 1965.
- (4) F. W. Smith, "Pattern Classifier Design by Linear Programming" IEEE Trans. on Computers, vol. C-17, No. 4, pp. 367-372, April 1968.
- (5) S. Muroga, Threshold Logic, Lecture notes for EE 497 and EE 498, Department of Computer Science, University of Illinois, 1965-1966.
- (6) S. Muroga, "Majority Logic and Problems of Probabilistic Behavior", in Self-Organizing Systems, Spartan Books, 1962, pp. 243-281.
- (7) A. J. Goldman, and A. W. Tucker, "Theory of Linear Programming", in Linear Inequalities and Related Systems, edited by H. W. Kuhn and A. W. Tucker, Princeton University Press, 1956, pp. 53-97.
- (8) G. B. Dantzig, Linear Programming and Extensions, Princeton University Press, Princeton, New Jersey, 1963.
- (9) G. Hadley, Linear Programming, Addison-Wesley, Addison-Wesley Series in Industrial Management, 1962.
- (10) R. E. Gomory, "An All-integer Integer Programming Algorithm" in Industrial Scheduling, Prentice-Hall International Series in Management, edited by Muth and Thompson, Prentice-Hall, 1963, pp. 193-206.

WAA

JAN 29 1973



UNIVERSITY OF ILLINOIS-URBANA



3 0112 002022835